Reactive Turing Machines

Baeten, J.C.M.; Luttik, B.; Tilburg van, P.J.A.

published in
Lecture Notes in Computer Science
2011

DOI (link to publisher)
10.1007/978-3-642-22953-4_30

document version
Publisher's PDF, also known as Version of record

Reactive Turing Machines

Jos C.M. Baeten, Bas Luttik, and Paul van Tilburg

Eindhoven University of Technology
{j.c.m.baeten,s.p.luttik,p.j.a.v.tilburg}@tue.nl

Abstract. We propose reactive Turing machines (RTMs), extending classical Turing machines with a process-theoretical notion of interaction. We show that every effective transition system is simulated modulo branching bisimilarity by an RTM, and that every computable transition system with a bounded branching degree is simulated modulo divergence-preserving branching bisimilarity. We conclude from these results that the parallel composition of (communicating) RTMs can be simulated by a single RTM. We prove that there exist universal RTMs modulo branching bisimilarity, but these essentially employ divergence to be able to simulate an RTM of arbitrary branching degree. We also prove that modulo divergence-preserving branching bisimilarity there are RTMs that are universal up to their own branching degree. Finally, we establish a correspondence between RTMs and the process theory TCP_τ.

1 Introduction

The Turing machine [19] is widely accepted as a computational model suitable for exploring the theoretical boundaries of computing. Motivated by the existence of universal Turing machines, many textbooks on the theory of computation present the Turing machine not just as a theoretical model to explain which functions are computable, but as an accurate conceptual model of the computer. There is, however, a well-known limitation to this view. A Turing machine operates from the assumptions that: (1) all input it needs for the computation is available on the tape from the very beginning; (2) it performs a terminating computation; and (3) it leaves the output on the tape at the very end. Thus, the notion of Turing machine abstracts from two key ingredients of computing: *interaction* and *non-termination*. Nowadays, most computing systems are so-called *reactive systems* [13], systems that are generally not meant to terminate and consist of computing devices that interact with each other and with their environment.

Concurrency theory emerged from the early work of Petri [16] and has now developed into a mature theory of reactive systems. We mention three of its contributions particularly relevant for our work. Firstly, it installed the notion of transition system as the prime mathematical model to represent discrete behaviour. Secondly, it offered the insight that language equivalence is too coarse in a setting with interacting automata; instead one should consider automata up to some form of bisimilarity. Thirdly, it yielded many algebraic process calculi facilitating the formal specification and verification of reactive systems.

O. Owe, M. Steffen and J.A. Telle (Eds.): FCT 2011, LNCS 6914, pp. 348–359, 2011.

In this paper we propose a notion of *reactive* Turing machine (RTM), extending the classical notion of Turing machine with interaction in the style of concurrency theory. The extension consists of a facility to declare every transition to be either *observable*, by labelling it with an action symbol, or unobservable, by labelling it with τ. Typically, a transition labelled with an action symbol models an interaction of the RTM with its environment (or some other RTM), while a transition labelled with τ refers to an internal computation step. Thus, a conventional Turing machine can be regarded as a special kind of RTM in which all transitions are declared unobservable by labelling them with τ.

The semantic object associated with a conventional Turing machine is either the function that it computes, or the formal language that it accepts. The semantic object associated with an RTM is a behaviour, formally represented by a transition system. A function is said to be effectively computable if it can be computed by a Turing machine. By analogy, we say that a behaviour is effectively executable if it can be exhibited by an RTM. In concurrency theory, behaviours are usually considered modulo a suitable behavioural equivalence. In this paper we shall mainly use *(divergence-preserving) branching bisimilarity* [11], which is the finest behavioural equivalence in Van Glabbeek's spectrum (see [9]).

In Sect. 3 we set out to investigate the expressiveness of RTMs up to divergence-preserving branching bisimilarity. We establish that every computable transition system with a bounded branching degree can be simulated, up to divergence-preserving branching bisimilarity, by an RTM. If the divergence-preservation requirement is dropped, even every effective transition system can be simulated. These results will then allow us to conclude that the behaviour of a parallel composition of RTMs can be simulated on a single RTM.

In Sect. 4 we define a suitable notion of universality for RTMs and investigate the existence of universal RTMs. We shall find that there are some subtleties pertaining to the branching degree bound associated with each RTM. Up to divergence-preserving branching bisimilarity, an RTM can at best simulate other RTMs with the same or a lower bound on their branching degree. If divergence-preservation is not required, however, then universal RTMs do exist.

In Sect. 5, we consider the correspondence between RTMs and the process theory TCP_τ. We establish that every executable behaviour is, again up to divergence-preserving branching bisimilarity, definable by a finite recursive TCP_τ-specification [1]. Recursive specifications are the concurrency-theoretic counterparts of grammars in the theory of formal languages. Thus, the result in Sect. 5 may be considered as the process-theoretic version of the correspondence between Turing machines and unrestricted grammars.

Several extensions of Turing machines with some form of interaction have been proposed in the literature, already by Turing in [20], and more recently in [6,12,21]. The goal in these works is mainly to investigate to what extent interaction may have a beneficial effect on the power of sequential computation. The focus remains on the computational aspect, and interaction is not treated as a first-class citizen. Our goal, instead, is to achieve integration of automata and concurrency theory that treats computation and interactivity on equal footing.

2 Reactive Turing Machines

We fix a finite set $\mathcal{A}$ of *action symbols* that we shall use to denote the observable events of a system. An unobservable event will be denoted with τ, assuming that $\tau \notin \mathcal{A}$; we shall henceforth denote the set $\mathcal{A} \cup \{\tau\}$ by $\mathcal{A}_\tau$. We also fix a finite set $\mathcal{D}$ of *data symbols*. We add to $\mathcal{D}$ a special symbol $\Box$ to denote a blank tape cell, assuming that $\Box \notin \mathcal{D}$; we denote the set $\mathcal{D} \cup \{\Box\}$ of *tape symbols* by $\mathcal{D}_\Box$.

Definition 1. *A* reactive Turing machine *(RTM) $\mathcal{M}$ is a quadruple $(\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ consisting of a finite set of* states $\mathcal{S}$*, a distinguished* initial state $\uparrow \in \mathcal{S}$*, a subset of* final states $\downarrow \subseteq \mathcal{S}$*, and a $(\mathcal{D}_\Box \times \mathcal{A}_\tau \times \mathcal{D}_\Box \times \{L, R\})$-labelled transition relation*

$$\rightarrow \subseteq \mathcal{S} \times \mathcal{D}_\Box \times \mathcal{A}_\tau \times \mathcal{D}_\Box \times \{L, R\} \times \mathcal{S} \ .$$

An RTM is deterministic *if $(s, d, a, e_1, M_1, t_1) \in \rightarrow$ and $(s, d, a, e_2, M_2, t_2) \in \rightarrow$ implies that $e_1 = e_2$, $t_1 = t_2$ and $M_1 = M_2$ for all $s, t_1, t_2 \in \mathcal{S}$, $d, e_1, e_2 \in \mathcal{D}_\Box$, $a \in \mathcal{A}_\tau$, and $M_1, M_2 \in \{L, R\}$, and, moreover, $(s, d, \tau, e_1, M_1, t_1) \in \rightarrow$ implies that there do not exist $a \neq \tau$, e_2, M_2, t_2 such that $(s, d, a, e_2, M_2, t_2) \in \rightarrow$*

If $(s, d, a, e, M, t) \in \rightarrow$, we write $s \xrightarrow{a[d/e]M} t$. The intuitive meaning of such a transition is that whenever $\mathcal{M}$ is in state s and d is the symbol currently read by the tape head, then it may execute the action a, write symbol e on the tape (replacing d), move the read/write head one position to the left or one position to the right on the tape (depending on whether $M = L$ or $M = R$), and then end up in state t. RTMs extend conventional Turing machines by associating with every transition an element $a \in \mathcal{A}_\tau$. The symbols in $\mathcal{A}$ are thought of as denoting observable activities; a transition labelled with an action symbol in $\mathcal{A}$ will semantically be treated as observable. Observable transitions are used to model interactions of an RTM with its environment or some other RTM, as will be explained more in detail below when we introduce a notion of parallel composition for RTMs. The symbol τ is used to declare that a transition is unobservable. A classical Turing machine is an RTM in which all transitions are declared unobservable.

Example 1. Assume that $\mathcal{A} = \{c!d, c?d \mid c \in \{i, o\} \ \&\ d \in \mathcal{D}_\Box\}$. Intuitively, i and o are the input/output communication channels by which the RTM can interact with its environment. The action symbol $c!d$ $(c \in \{i, o\})$ then denotes the event that a datum d is sent by the RTM along channel c, and the action symbol $c?d$ $(c \in \{i, o\})$ denotes the event that a datum d is received by the RTM along channel c.

The left state-transition diagram in Fig. 1 specifies an RTM that first inputs a string, consisting of an arbitrary number of 1s followed by the symbol #, stores the string on the tape, and returns to the beginning of the string. Then, it performs a computation to determine if the number of 1s is odd or even. In the first case, it simply removes the string from the tape and returns to the initial state. In the second case, it outputs the entire string, removes it from the tape, and returns to the initial state. The right state-transition diagram in Fig. 1 outputs on channel i the infinite sequence $1\#11\#111\#\ldots\#1^n\#\ldots$ $(n \geq 1)$.

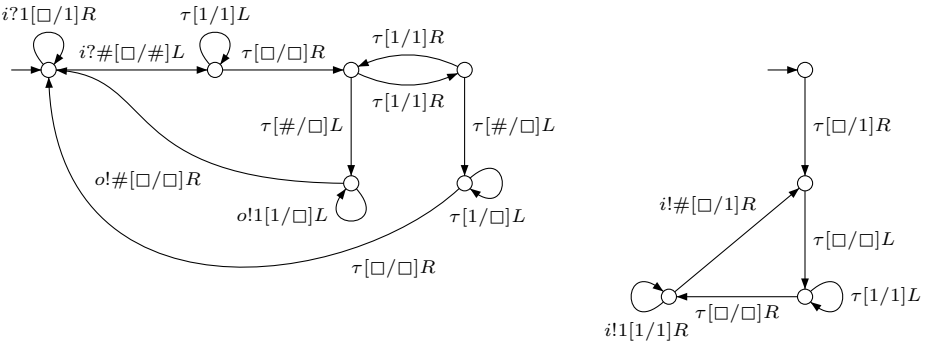

Fig. 1. Examples of reactive Turing machines

To formalise our intuitive understanding of the operational behaviour of RTMs we shall below associate with every RTM a transition system. An $\mathcal{A}_\tau$*-labelled transition system* T is a quadruple $(\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ consisting of a set of *states* $\mathcal{S}$, an *initial* state $\uparrow \in \mathcal{S}$, a subset $\downarrow \subseteq \mathcal{S}$ of *final* states, and an $\mathcal{A}_\tau$-labelled *transition relation* $\rightarrow \subseteq \mathcal{S} \times \mathcal{A}_\tau \times \mathcal{S}$. If $(s, a, t) \in \rightarrow$, we write $s \xrightarrow{a} t$. If s is a final state, i.e., $s \in \downarrow$, we write $s\!\downarrow$. The transition system T is *deterministic* if, for every state $s \in \mathcal{S}$ and for every $a \in \mathcal{A}_\tau$, $s \xrightarrow{a} s_1$ and $s \xrightarrow{a} s_2$ implies $s_1 = s_2$, and, moreover, $s \xrightarrow{\tau} s_1$ implies that there do not exist an action $a \neq \tau$ and a state s_2 such that $s \xrightarrow{a} s_2$.

With every RTM $\mathcal{M}$ we are going to associate a transition system $\mathcal{T}(\mathcal{M})$. The states of $\mathcal{T}(\mathcal{M})$ are the configurations of the RTM, consisting of a state of the RTM, its tape contents, and the position of the read/write head on the tape. We represent the tape contents by an element of $(\mathcal{D}_\Box)^*$, replacing precisely one occurrence of a tape symbol d by a *marked* symbol $\check{d}$, indicating that the read/write head is on this symbol. We denote by $\check{\mathcal{D}}_\Box = \{\check{d} \mid d \in \mathcal{D}_\Box\}$ the set of *marked* tape symbols; a *tape instance* is a sequence $\delta \in (\mathcal{D}_\Box \cup \check{\mathcal{D}}_\Box)^*$ such that δ contains exactly one element of $\check{\mathcal{D}}_\Box$. Formally, a *configuration* is now a pair (s, δ) consisting of a state $s \in \mathcal{S}$, and a tape instance δ.

Our transition system semantics defines an $\mathcal{A}_\tau$-labelled transition relation on configurations such that an RTM-transition $s \xrightarrow{a[d/e]M} t$ corresponds with a-labelled transitions from configurations consisting of the RTM-state s and a tape instance in which some occurrence of d is marked. The transitions lead to configurations consisting of t and a tape instance in which the marked symbol d is replaced by e, and either the symbol to the left or to right of this occurrence of e is replaced by its marked version, according to whether $M = L$ or $M = R$. If e happens to be the first symbol and $M = L$, or the last symbol and $M = R$, then an additional blank symbol is appended at the left or right end of the tape instance, respectively, to model the movement of the head.

We introduce some notation to concisely denote the new placement of the tape head marker. Let δ be an element of $\mathcal{D}_\Box^*$. Then by $\delta^<$ we denote the element of $(\mathcal{D}_\Box \cup \check{\mathcal{D}}_\Box)^*$ obtained by placing the tape head marker on the right-most symbol of δ if it exists, and $\check{\Box}$ otherwise. Similarly, by ${}^>\delta$ we denote the element

of $(\mathcal{D}_\Box \cup \check{\mathcal{D}}_\Box)^*$ obtained by placing the tape head marker on the left-most symbol of δ if it exists, and $\check{\Box}$ otherwise.

Definition 2. *Let $\mathcal{M} = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ be an RTM. The* transition system $\mathcal{T}(\mathcal{M})$ associated with $\mathcal{M}$ *is defined as follows:*

1. *its set of states is the set of all configurations of $\mathcal{M}$;*
2. *its transition relation $\rightarrow$ is the least relation satisfying, for all $a \in \mathcal{A}_\tau$, $d, e \in \mathcal{D}_\Box$ and $\delta_L, \delta_R \in \mathcal{D}_\Box^*$:*

$$(s, \delta_L \check{d} \delta_R) \xrightarrow{a} (t, \delta_L {}^{<}e \delta_R) \text{ iff } s \xrightarrow{a[d/e]L} t \text{ , and}$$
$$(s, \delta_L \check{d} \delta_R) \xrightarrow{a} (t, \delta_L e {}^{>}\delta_R) \text{ iff } s \xrightarrow{a[d/e]R} t \text{ ;}$$

3. *its initial state is the configuration $(\uparrow, \check{\Box})$; and*
4. *its set of final states is the set of terminating configurations $\{(s, \delta) \mid s\downarrow\}$.*

Turing introduced his machines to define the notion of *effectively computable function*. By analogy, our notion of RTM can be used to define a notion of *effectively executable behaviour*.

Definition 3. *A transition system is* executable *if it is associated with an RTM.*

Parallel composition. To illustrate how RTMs are suitable to model a form of interaction, we shall now define on RTMs a notion of parallel composition, equipped with a simple form of communication. Let $\mathcal{C}$ be a finite set of *channels* for the communication of data symbols between one RTM and another. Intuitively, $c!d$ stands for the action of sending datum d along channel c, while $c?d$ stands for the action of receiving datum d along channel c.

First, we define a notion of parallel composition on transition systems. Let $T_1 = (\mathcal{S}_1, \rightarrow_1, \uparrow_1, \downarrow_1)$ and $T_2 = (\mathcal{S}_2, \rightarrow_2, \uparrow_2, \downarrow_2)$ be transition systems, and let $\mathcal{C}' \subseteq \mathcal{C}$. The *parallel composition* of T_1 and T_2 is the transition system $[T_1 \parallel T_2]_{\mathcal{C}'} = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$, with $\mathcal{S}$, $\rightarrow$, $\uparrow$ and $\downarrow$ defined by

1. $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2$;
2. $(s_1, s_2) \xrightarrow{a} (s_1', s_2')$ iff $a \in \mathcal{A}_\tau - \{c!d, c?d \mid c \in \mathcal{C}', d \in \mathcal{D}_\Box\}$ and either
   (a) $s_1 \xrightarrow{a} s_1'$ and $s_2 = s_2'$, or $s_2 \xrightarrow{a} s_2'$ and $s_1 = s_1'$, or
   (b) $a = \tau$ and either $s_1 \xrightarrow{c!d} s_1'$ and $s_2 \xrightarrow{c?d} s_2'$, or $s_1 \xrightarrow{c?d} s_1'$ and $s_2 \xrightarrow{c!d} s_2'$ for some $c \in \mathcal{C}'$ and $d \in \mathcal{D}_\Box$;
3. $\uparrow = (\uparrow_1, \uparrow_2)$; and
4. $\downarrow = \{(s_1, s_2) \mid s_1 \in \downarrow_1 \;\&\; s_2 \in \downarrow_2\}$.

Definition 4. *Let $\mathcal{M}_1 = (\mathcal{S}_1, \rightarrow_1, \uparrow_1, \downarrow_1)$ and $\mathcal{M}_2 = (\mathcal{S}_2, \rightarrow_2, \uparrow_2, \downarrow_2)$ be RTMs, and let $\mathcal{C}' \subseteq \mathcal{C}$; by $[\mathcal{M}_1 \parallel \mathcal{M}_2]_{\mathcal{C}'}$ we denote the* parallel composition *of $\mathcal{M}_1$ and $\mathcal{M}_2$. The transition system $\mathcal{T}([\mathcal{M}_1 \parallel \mathcal{M}_2]_{\mathcal{C}'})$ associated with the parallel composition $[\mathcal{M}_1 \parallel_{\mathcal{C}} \mathcal{M}_2]_{\mathcal{C}'}$ of $\mathcal{M}_1$ and $\mathcal{M}_2$ is the parallel composition of the transition systems associated with $\mathcal{M}_1$ and $\mathcal{M}_2$, i.e., $\mathcal{T}([\mathcal{M}_1 \parallel \mathcal{M}_2]_{\mathcal{C}'}) = [\mathcal{T}(\mathcal{M}_1) \parallel \mathcal{T}(\mathcal{M}_2)]_{\mathcal{C}'}$.*

Example 2. Let $\mathcal{A}$ be as in Example 1, let $\mathcal{M}$ denote the left-hand side RTM in Fig. 1, and let $\mathcal{E}$ denote the right-hand side RTM in Fig. 1. Then the parallel composition $[\mathcal{M} \parallel \mathcal{E}]_i$ exhibits the behaviour of outputting, along channel o, the string $11\#1111\#\cdots\#1^n\#$ ($n \geq 2$, n even).

Equivalences. In automata theory, Turing machines that compute the same function or accept the same language are generally considered equivalent. In fact, functional or language equivalence is underlying many of the standard notions and results in automata theory. Perhaps most notably, a *universal* Turing machine is a Turing machine that, when started with the code of some Turing machine on its tape, simulates this machine up to functional or language equivalence. A result from concurrency theory is that functional and language equivalence are arguably too coarse for reactive systems, because they abstract from all moments of choice (see, e.g., [1]). In concurrency theory many alternative behavioural equivalences have been proposed; we refer to [9] for a classification.

The results about RTMs that are obtained in the remainder of this paper are modulo *branching bisimilarity* [11]. We shall consider both the divergence-insensitive and the divergence-preserving variant. Let T_1 and T_2 be transition systems. If T_1 and T_2 are *branching bisimilar*, then we write $T_1 \leftrightarrows_{\mathrm{b}} T_2$. If T_1 and T_2 are *divergence-preserving branching bisimilar*, then we write $T_1 \leftrightarrows_{\mathrm{b}}^{\Delta} T_2$. (Due to space limitations, the formal definitions had to be omitted; the reader is referred to the full version [4], or to [10], where the divergence-preserving variant is called *branching bisimilarity with explicit divergence*.)

3 Expressiveness of RTMs

We shall establish in this section that every effective transition system can be simulated by an RTM up to branching bisimilarity, and that every boundedly branching computable transition system can be simulated up to divergence-preserving branching bisimilarity. We use this as an auxiliary result to establish that a parallel composition of RTMs can be simulated by a single RTM.

Let $T = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ be a transition system; the mapping $out : \mathcal{S} \rightarrow 2^{\mathcal{A}_\tau \times \mathcal{S}}$ associates with every state its set of outgoing transitions, i.e., for all $s \in \mathcal{S}$, $out(s) = \{(a,t) \mid s \xrightarrow{a} t\}$, and $fin(\_)$ denotes the characteristic function of $\downarrow$.

Definition 5. *Let $T = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ be an $\mathcal{A}_\tau$-labelled transition system. We say that T is* effective *if there exist suitable codings of $\mathcal{A}_\tau$ and $\mathcal{S}$ (into the natural numbers) such that $\rightarrow$ and $\downarrow$ are recursively enumerable. We say that T is* computable *if there exist suitable codings of $\mathcal{A}_\tau$ and $\mathcal{S}$ such that the functions $out(\_)$ and $fin(\_)$ are recursive.*

The notion of effective transition system originates with Boudol [7]. For the notion of computable transition system we adopt the definition from [2]. If $\rightarrow$ and $\downarrow$ are recursively enumerable, then there exist algorithms that enumerate the transitions in $\rightarrow$ and the states in $\downarrow$. If the functions $out(\_)$ and $fin(\_)$ are recursive, then there exists an algorithm that, given a state s, yields the list of outgoing transitions of s and determines if $s \in \downarrow$.

Proposition 1. *The transition system associated with an RTM is computable.*

Hence, unsurpisingly, if a transition system is not computable, then it is not executable either. It is easy to define transition systems that are not computable, so there exist behaviours that are not executable. The full version of this paper [4] contains an example that illustrates that there exist behaviours that are not even executable up to branching bisimilarity.

Phillips associates, in [17], with every effective transition system a *branching bisimilar* computable transition system of which, moreover, every state has a branching degree of at most 2. (Phillips actually establishes weak bisimilarity, but it is easy to see that branching bisimilarity holds.)

Definition 6. *Let $T = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ be a transition system, and let B be a natural number. We say that T has a branching degree* bounded by B *if, for every state $s \in \mathcal{S}$, $|out(s)| \leq B$. We say that T is* boundedly branching *if there exists $B \in \mathbb{N}$ such that the branching degree of T is bounded by B.*

Proposition 2 (Phillips). *For every effective transition system T there exists a boundedly branching computable transition system T' such that $T \leftrightarrows_b T'$.*

A crucial insight in Phillips' proof is that a divergence (i.e., an infinite sequence of τ-transitions) can be exploited to simulate a state of which the set of outgoing transitions is recursively enumerable, but not recursive. The following example, inspired by [8], shows that introducing divergence is unavoidable.

Example 3. (In this and later examples, we denote by φ_x the partial recursive function with index $x \in \mathbb{N}$ in some exhaustive enumeration of partial recursive functions, see, e.g., [18].) Assume that $\mathcal{A} = \{a, b\}$, and consider the transition system $T_1 = (\mathcal{S}_1, \rightarrow_1, \uparrow_1, \downarrow_1)$ with $\mathcal{S}_1$, $\rightarrow_1$, $\uparrow_1$ and $\downarrow_1$ defined by

$$\mathcal{S}_1 = \{s_{1,x},\ t_{1,x} \mid x \in \mathbb{N}\}\ ,\quad \uparrow_1 = s_{1,0}\ ,$$
$$\rightarrow_1 = \{(s_{1,x}, a, s_{1,x+1}) \mid x \in \mathbb{N}\} \cup \{(s_{1,x}, b, t_{1,x}) \mid x \in \mathbb{N}\}\ ,\ \text{and}$$
$$\downarrow_1 = \{t_{1,x} \mid \varphi_x(x) \text{ converges}\}\ .$$

If T_2 is a transition system such that $T_1 \leftrightarrows_{\mathrm{b}}^{\Delta} T_2$, as witnessed by some divergence-preserving branching bisimulation relation $\mathcal{R}$, then it can be argued that T_2 is not computable. A detailed argument can be found in the full version [4].

By Proposition 2, in order to prove that every effective transition system can be simulated up to branching bisimilarity by an RTM, it suffices to prove that every boundedly branching computable transition system can be simulated by an RTM. Let $T = (\mathcal{S}_T, \rightarrow_T, \uparrow_T, \downarrow_T)$ be a boundedly branching computable transition system, say with branching degree bounded by B. It is reasonably straightforward to construct an RTM $\mathcal{M} = (\mathcal{S}_\mathcal{M}, \rightarrow_\mathcal{M}, \uparrow_\mathcal{M}, \downarrow_\mathcal{M})$, which we call the *simulator* for T, such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_{\mathrm{b}}^{\Delta} T$. The construction is detailed in [4].

Theorem 1. *For every boundedly branching computable transition system T there exists an RTM $\mathcal{M}$ such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^{\Delta} T$.*

Combining Theorem 1 with Proposition 2 we can conclude that RTMs can simulate effective transition systems up to branching bisimilarity, but, in view of Example 3, not in a divergence-preserving manner.

Corollary 1. *For every effective transition system T there exists a reactive Turing machine $\mathcal{M}$ such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b T$.*

All computations involved in the simulation of T are deterministic; if $\mathcal{M}$ is nondeterministic, then this is due to a state of which the menu includes some action a more than once. Clearly, if T is deterministic, then, for every state s in T, $|out(s)| \leq |\mathcal{A}_\tau|$. So a deterministic transition system is boundedly branching, and therefore we get the following corollary to Theorem 1.

Corollary 2. *For every deterministic computable transition system T there exists a deterministic RTM $\mathcal{M}$ such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^\Delta T$.*

Using Theorem 1 we can now also establish that a parallel composition of RTMs can be simulated, up to divergence-preserving branching bisimilarity, by a single RTM. To this end, note that the transition systems associated with RTMs are boundedly branching and computable. Further note that the parallel composition of boundedly branching computable transition systems is again computable. It follows that the transition system associated with a parallel composition of RTMs is boundedly branching and computable, and hence, by Theorem 1, there exists an RTM that simulates this transition system up to divergence-preserving branching bisimilarity. Thus we get the following corollary.

Corollary 3. *For every pair of RTMs $\mathcal{M}_1$ and $\mathcal{M}_2$ and for every set of communication channels $\mathcal{C}$ there is an RTM $\mathcal{M}$ such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^\Delta \mathcal{T}([\mathcal{M}_1 \parallel \mathcal{M}_2]_\mathcal{C})$.*

4 Universality

Recall that a *universal Turing machine* is some Turing machine that can simulate an arbitrary Turing machine on arbitrary input. The assumptions are that both the finite description of the to be simulated Turing machine and its input are available on the tape of the universal Turing machine, and the simulation is up to functional or language equivalence. We adapt this scheme in two ways. Firstly, we let the simulation start by inputting the description of an arbitrary RTM $\mathcal{M}$ along some dedicated channel u, rather than assuming its presence on the tape right from the start. This is both conceptually desirable —for our aim is to give interaction a formal status—, and technically necessary —for in the semantics of RTMs we have assumed that the tape is initially empty. Secondly, we require the behaviour of $\mathcal{M}$ to be simulated up to divergence-preserving branching bisimilarity.

Thus, we arrive at the following tentative definitions. For an arbitrary RTM $\mathcal{M}$, denote by $\overline{\mathcal{M}}$ an RTM that outputs a description of $\mathcal{M}$ along channel u and then terminates. A *universal* RTM is then an RTM $\mathcal{U}$ such that, for every RTM $\mathcal{M}$, the parallel composition $\left[\mathcal{U} \parallel \overline{\mathcal{M}}\right]_{\{u\}}$ simulates $\mathcal{T}(\mathcal{M})$.

Although such a universal RTM $\mathcal{U}$ exists up to branching bisimilarity as we shall see below, it does not exist up to divergence-preserving branching bisimilarity. To see this, note that the transition system associated with any particular RTM $\mathcal{U}$ has a branching degree that is bounded by some natural number B. It can then be established that, up to divergence-preserving branching bisimilarity, $\mathcal{U}$ can only simulate RTMs with a branching degree bounded by B. The argument is formalised in the following proposition; see [4] for a proof.

Proposition 3. *There does not exist an RTM $\mathcal{U}$ such that for all RTM $\mathcal{M}$ it holds that $\left[\mathcal{U} \parallel \overline{\mathcal{M}}\right]_{\{u\}} \leftrightarrows_b^{\Delta} \mathcal{T}(\mathcal{M})$.*

If we insist on simulation up to divergence-preserving branching bisimilarity, then we need to relax the notion of universality. Let B be a natural number. An RTM $\mathcal{U}_B$ is *universal up to* B if for every RTM $\mathcal{M}$ with $\mathcal{T}(\mathcal{M})$ bounded by branching degree B it holds that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^{\Delta} \left[\overline{\mathcal{M}} \parallel \mathcal{U}_B\right]_{\{u\}}$.

The construction of the simulator for a transition system of which the branching degree is bounded by B in the proof of Theorem 1 can be adapted to get the definition of an RTM $\mathcal{U}_B$ that is universal up to B. It suffices to slightly modify the initialisation fragment. Instead of writing the codes of the functions $out(\_)$ and $fin(\_)$ and the initial state directly on the tape, the *initialisation fragment* receives the code $\ulcorner \mathcal{M} \urcorner$ of an arbitrary $\mathcal{M}$ along some dedicated channel u. Then, it recursively computes the codes of the functions $out(\_)$ and $fin(\_)$, and the initial state of $\mathcal{T}(\mathcal{M})$ and stores these on the tape.

Theorem 2. *For every B there exists an RTM $\mathcal{U}_B$ such that, for all RTMs $\mathcal{M}$ with a branching degree bounded by B, it holds that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^{\Delta} \left[\overline{\mathcal{M}} \parallel \mathcal{U}_B\right]_{\{u\}}$.*

If we drop divergence-preservation as a requirement for the simulation, then a universal RTM does exist. At the heart of the argument is a trick, first described in [2] and adapted by Phillips in [17], to use a divergence with (infinitely many) states of at most a branching degree of 2 to simulate, up to branching bisimilarity, a single state of some arbitrary (even countably infinite) branching degree.

Theorem 3. *There exists an RTM $\mathcal{U}$ such that, for all RTMs $\mathcal{M}$, it holds that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b \left[\mathcal{U} \parallel \overline{\mathcal{M}}\right]_{\{u\}}$.*

5 Recursive Specifications

A well-known result from the theory of automata and formal languages is that the formal languages accepted by Turing machines correspond with the languages generated by an unrestricted grammar. A *grammar* is a formal system for describing a formal language. The corresponding notion in concurrency theory is the notion of *recursive specification*, which is a formal system for describing behaviour. In this section, we show that the behaviours of RTMs correspond with the behaviours described by so-called TCP_τ recursive specifications. The process theory TCP_τ is a general theory for describing behaviour, encompassing the key features of the well-known process theories ACP_τ [5], CCS [15] and CSP [14].

We shall briefly introduce the syntax of TCP_τ and informally describe its operational semantics. We refer to the textbook [1] for an elaborate treatment. We reuse the finite set $\mathcal{C}$ of channels and set of data $\mathcal{D}_\Box$ introduced in Sect. 2; we introduce the set of special actions $\mathcal{I} = \{c?d, c!d \mid d \in \mathcal{D}_\Box, c \in \mathcal{C}\}$. The actions $c?d$ and $c!d$ denote the events that a datum d is received or sent along *channel* c. Let $\mathcal{N}$ be a countably infinite set of names. The set of *process expressions* $\mathcal{P}$ is generated by the following grammar ($a \in \mathcal{A}_\tau \cup \mathcal{I}, N \in \mathcal{N}, \mathcal{C}' \subseteq \mathcal{C}$):

$$p \ ::= \ \mathbf{0} \mid \mathbf{1} \mid a.p \mid p \cdot p \mid p + p \mid [p \parallel p]_{\mathcal{C}'} \mid N \ .$$

The constant $\mathbf{0}$ denotes *deadlock*, the unsuccessfully terminated process. The constant $\mathbf{1}$ denotes *skip*, the successfully terminated process. For each action $a \in \mathcal{A} \cup \mathcal{I}$ there is a unary operator $a.$ denoting action prefix; the process denoted by $a.p$ can do an a-transition to the process denoted by p. The binary operator $\cdot$ denotes *sequential composition*. The binary operator $+$ denotes *alternative composition* or *choice*. The binary operator $[\_ \parallel \_]_{\mathcal{C}'}$ denotes the special kind of *parallel composition* that we have also defined on RTMs. It enforces communication along the channels in $\mathcal{C}'$, and communication results in τ. (By including the restricted kind of parallel composition, we deviate from the definition of TCP_τ discussed in [1], but we note that our notion of parallel composition is definable with the operations $\parallel$, $\partial_\_(\_)$ and $\tau_\_(\_)$ of TCP_τ in [1].)

A *recursive specification* E is a set of equations of the form: $N \stackrel{\text{def}}{=} p$, with as left-hand side a name N and as right-hand side a TCP_τ process expression p. It is required that a recursive specification E contains, for every $N \in \mathcal{N}$, at most one equation with N as left-hand side; this equation will be referred to as the *defining equation* for N in $\mathcal{N}$. Furthermore, if some name occurs in the right-hand side of some defining equation, then the recursive specification must include a defining equation for it. Let E be a recursive specification, and let p be a process expression. There is a standard method to associate with p a transition system $\mathcal{T}_E(p)$. The details can be found, e.g., in [1].

In [4] we present the details of a construction that associates with an arbitrary RTM a TCP_τ recursive specification that defines its behaviour up to divergence-preserving branching bisimilarity. Thus, we get the following correspondence.

Theorem 4. *For every RTM $\mathcal{M}$ there exists a finite recursive specification E and a process expression p such that $\mathcal{T}(\mathcal{M}) \leftrightarrows_b^\Delta \mathcal{T}_E(p)$,*

As a corollary we find that every executable transition system is definable, up to divergence-preserving branching bisimilarity, by a TCP_τ recursive specification. Since there exist recursive specifications with an unboundedly branching associated transition system (see, e.g., [3], for the converse of the abovementioned theorem), we have to give up divergence-preservation. Since the transition system associated with a finite recursive specification is clearly effective, we do get, by Corollary 1, the following result.

Corollary 4. *For every finite recursive specification E and process expression p, there exists an RTM $\mathcal{M}$ such that $\mathcal{T}_E(p) \leftrightarrows_b \mathcal{T}(\mathcal{M})$.*

6 Concluding Remarks and Future Work

We have proposed a notion of reactive Turing machine and discussed its expressiveness in bisimulation semantics. Although it is not the aim of this work to contribute to the debate as to whether interactive computation is more powerful than traditional computation, our notion of RTM may nevertheless turn out to be a useful tool in the discussion. For instance, our result that the parallel composition of RTMs can be simulated by an RTM seems to contradict the implied conjecture in [12, Sect. 11] that interactive computation performed by multiple machines working together is more expressive than interactive computation performed by a single machine.

To be sure, however, we would need to firmly establish the robustness of our notion by showing that variations on its definition (e.g., multiple tracks or multiple tapes), and by showing that it can simulate the other proposals (persistent Turing machines [12], interactive Turing machines [21]). We also intend to consider interactive versions of other computational models. The λ-calculus would be an interesting candidate to consider, because of the well-known result that it is inherently sequential; this suggests that an interactive version of λ-calculus will be less expressive than RTMs. In particular, we conjecture that the evaluation of *parallel-or* or McCarthy's *amb* can be simulated with RTMs.

RTMs may also prove to be a useful tool in establishing the expressiveness of process theories. For instance, the transition system associated with a π-calculus expression is effective, so it can be simulated by an RTM, at least up to branching bisimilarity. The π-calculus can to some extent be seen as the interactive version of the λ-calculus. We conjecture that the converse —every executable transition system can be specified by a π-calculus expression— is also true.

Petri showed already in his thesis [16] that concurrency and interaction may serve to bridge the gap between the theoretically convenient Turing machine model of a sequential machine with unbounded memory, and the practically more realistic notion of extendable architecture of components with bounded memory. The specification we present in the proof of Theorem 4 (see [4]) is another illustration of this idea: the unbounded tape is modelled as an unbounded parallel composition. It would be interesting to further study the inherent tradeoff between unbounded parallel composition and unbounded memory in the context of RTMs, considering unbounded parallel compositions of RTMs with bounded memory.

Acknowledgement. We thank Herman Geuvers for discussions, and Clemens Grabmayer for suggesting to us the term *reactive Turing machine*, and the anonymous referees for their useful comments.

References

1. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process Algebra (Equational Theories of Communicating Processes). Cambridge University Press, Cambridge (2009)
2. Baeten, J.C.M., Bergstra, J.A., Klop, J.W.: On the consistency of Koomen's fair abstraction rule. Theoretical Computer Science 51, 129–176 (1987)

3. Baeten, J.C.M., Cuijpers, P.J.L., Luttik, B., van Tilburg, P.J.A.: A Process-Theoretic Look at Automata. In: Arbab, F., Sirjani, M. (eds.) FSEN 2009. LNCS, vol. 5961, pp. 1–33. Springer, Heidelberg (2010)
4. Baeten, J.C.M., Luttik, B., van Tilburg, P.J.A.: Reactive Turing machines. CoRR, abs/1104.1738 (2011)
5. Bergstra, J.A., Klop, J.W.: Algebra of communicating processes with abstraction. Theoretical Computer Science 37, 77–121 (1985)
6. Blass, A., Gurevich, Y., Rosenzweig, D., Rossman, B.: Interactive small-step algorithms I: Axiomatization. Logical Methods in Computer Science 3(4) (2007)
7. Boudol, G.: Notes on algebraic calculi of processes. In: Apt, K.R. (ed.) Logics and Models of Concurrent Systems. NATO-ASI Series, vol. F13, pp. 261–303. Springer, Berlin (1985)
8. Darondeau, P.: Bisimulation and effectiveness. Inf. Process. Lett. 30(1), 19–20 (1989)
9. van Glabbeek, R.J.: The Linear Time – Branching Time Spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
10. van Glabbeek, R.J., Luttik, B., Trcka, N.: Branching bisimilarity with explicit divergence. Fundam. Inform. 93(4), 371–392 (2009)
11. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM 43(3), 555–600 (1996)
12. Goldin, D.Q., Smolka, S.A., Attie, P.C., Sonderegger, E.L.: Turing machines, transition systems, and interaction. Inf. Comput. 194(2), 101–128 (2004)
13. Harel, D., Pnueli, A.: On the development of reactive systems. In: Apt, K.R. (ed.) Logics and Models of Concurrent Systems. NATO ASI Series, vol. F-13, pp. 477–498. Springer, New York (1985)
14. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
15. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
16. Petri, C.A.: Kommunikation mit Automaten. PhD thesis, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2 (1962)
17. Phillips, I.C.C.: A note on expressiveness of process algebra. In: Burn, G.L., Gay, S., Ryan, M.D. (eds.) Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods, Workshops in Computing, pp. 260–264. Springer, Heidelberg (1993)
18. Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill Book Company, New York (1967); Reprinted by MIT Press (1987)
19. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 2(42), 230–265 (1936)
20. Turing, A.M.: Systems of logic based on ordinals. Proceedings of the London Mathematical Society 45(2), 161–228 (1939)
21. van Leeuwen, J., Wiedermann, J.: On algorithms and interaction. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 99–113. Springer, Heidelberg (2000)